

MECHANISM FOR EFFICIENTLY PROCESSING DEFERRED ORDER- DEPENDENT MEMORY ACCESS TRANSACTIONS IN A PIPELINED SYSTEM

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of pipelining memory accesses. In particular, the invention is related to pipelining ordered memory accesses.

2. Description of Related Art

As processor performance continues to outpace memory performance, reducing the latency in memory accesses is critical in achieving high performance in today's computer systems. One method to reduce this latency is pipelining. Pipelining is a mechanism in which several stages or phases in the processing of instructions are carried out in an overlapped or parallel manner such that overall instruction throughput is optimized.

In a typical prior art architecture, a bus activity is hierarchically organized into operations, transactions, and phases. An operation is a bus procedure that appears atomic to software even though it may not be atomic on the bus. A transaction is the set of bus activities related to a single bus request. A transaction may contain up to six phases. A phase uses a specific set of signals to communicate a particular type of information. The six phases of the prior art processor bus protocol are: arbitration, request, error, snoop, response, and data.

In the arbitration phase, a bus agent which is not the current bus owner requests the bus. In the request phase, the bus owner drives request and address information on the bus. In the error phase, any parity errors triggered by the request are reported. In the snoop phase, the address information is determined if it references a valid or modified (dirty) cache line. In addition, the snoop results also indicate whether a transaction will be completed in-order or may be deferred for possible out-of-order completion. In the response phase, the status of the transaction is reported (whether failure or success, whether completion is immediate or deferred, etc.). The data phase is needed when the bus agent requests a data transfer such as read or write.

Instructions are typically executed in order, i.e., in the same order that they are originally written. Therefore, the transactions initiated by the instructions are required to be ordered. If, for some reason, the transaction cannot be performed immediately, then it has to be deferred. There is a signal asserted by the system to inform the processor that the transaction has been deferred so that the processor is prevented from issuing further order-dependent transactions. Because the deferred transaction takes some time to be completed, the processor has to wait until it receives a signal stating the status of the deferred transaction before it can issue the next order-dependent transactions. This waiting time imposes a penalty to the overall system throughput, especially in high performance system.

It is, therefore, desirable to have a mechanism that allows the processor to continue issuing in-order transactions without waiting for the status of the deferred transactions.

SUMMARY OF THE INVENTION

The present invention describes a method and a system for efficient pipelining ordered memory accesses which includes transactions to be processed in order. A transaction is deferred if it cannot be completed in order. A visibility

signal is provided if a sequential order of the deferred transaction and the next ordered transaction is guaranteed.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

FIG. 1 is a block diagram illustration of one embodiment of a system that operates in accordance with the teachings of the present invention.

FIG. 2 is a table showing the various encodings of the snoop signals HIT#, HITM#, DEFER#, and VSBL#.

FIG. 3 is a flowchart illustrating the teachings of the present invention.

FIG. 4 is a timing diagram illustrating the relationship of the snoop signals and the defer signals.

DESCRIPTION OF THE PRESENT INVENTION

In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention unnecessarily.

In the following description, conventional signal names and symbols are used. For example, the symbol "#" represents an active low signal. In addition, the following definitions are relevant:

Agent: Any device that connects to the processor bus including the processors themselves. The term agent used here refers to "bus agent". There are several classifications of bus agents: central agent, input/output (I/O) agent, and memory agent. A central agent is one that handles reset, hardware configuration and initialization, special transactions, and centralized hardware error detection and handling. An I/O agent is one that interfaces to I/O devices using I/O port addresses; it can be a bus bridge to another bus used for I/O devices, such as a PCI bridge. A memory agent is one that provides access to main memory.

In-order execution: A type of execution that executes the transactions in the same order as they were issued.

Out-of-order execution: A type of execution that enables the processor to execute an instruction even if previous instructions in the execution stream have not completed or executed.

Speculative execution: A type of execution that enables the processor to execute an instruction that may or may not be part of the execution stream (such as an instruction following a conditional branch), so long as the processor can undo the instruction's effect if it is not part of the execution stream.

The present invention discloses a method and apparatus for efficient pipelining of ordered memory accesses on a parallel bus. Memory access transactions on a parallel bus can be deferred using a defer signal. A visibility guarantee signal is employed to indicate that a deferred transaction will not be retried and will be sequentially consistent in all subsequent transactions. The processor then can initiate the next ordered transaction without waiting for the report of the completion of the deferred transaction, resulting in fast overall throughput.

Referring to FIG. 1, an illustrative embodiment of a computer system utilizing the present invention is shown.

The computer system 100 comprises a number of processors 105₁–105_N (“N” being a positive whole number) and a main memory element 130 (e.g., dynamic random access memory “DRAM”, static random access memory “SRAM” etc.) coupled together by a chipset 120. The chipset 120 operates as an interface between a plurality of buses, namely a host bus 110, a peripheral bus 135 and an expansion bus 155.

Processors 105₁–105_N are any microprocessors. In this embodiment, processors 105₁–105_N are the Pentium® or Pentium® Pro™ microprocessors manufactured by Intel Corporation at Santa Clara, Calif.

Chipset 120 typically includes cache DRAM controller (“CDC”), peripheral bus (e.g., PCI) controller, and data path unit (“DPU”). The PCI Bridge (“PB”) provides a set of host-to-PCI and PCI-to-host bus transaction translations

The host bus 110 is a bus that can support transactions to a number of connected processors. Host bus 110 may be referred to as a parallel bus or multiprocessor bus because it supports parallel operations and multiple processors. It is contemplated that host bus 110 operates in a pipelined manner to increase efficiency. However, these features are not construed to limit the teachings of the present invention. The present invention can be utilized if there is only one processor connected to the host bus 110.

The peripheral bus 135 provides a communication path between the processors 105₁–105_N or main memory element 130 and a plurality of peripheral devices 150₁–150_P (“P” being a positive whole number). These peripheral devices 150₁–150_P may include I/O devices such as disk controller, data entry interface circuits, and pointing device controllers. The peripheral bus 135 may include a Peripheral Component Interconnect (“PCI”) bus or any other type of bus architecture.

The expansion bus 155 provides a communication path between the peripheral bus 135 and a plurality of expansion peripheral devices 160₁–160_K (“K” being a positive whole number). The expansion bus 155 may include an Industry Standard Architecture (“ISA”) bus or an Extended Industry Standard Architecture (“EISA”) bus.

Signals on Host Bus 110:

Bus activity is hierarchically organized into transactions and phases. A transaction is comprised of multiple phases. Most phases occur in order. The host bus 110 pipelines transaction in which bus transactions in different phases overlap.

One embodiment of the present invention explicitly defines a Deferred phase. A transaction enters the Deferred phase or a Deferred Reply transaction if it has been deferred in the Snoop and Response phases and the transaction is ready for completion. A Data phase is possible. The agent responsible for the Deferred phase or the Deferred Reply transaction is usually the responding agent and is referred to as the deferring agent. After completion of the Deferred phase or the Deferred Reply transaction, the transaction has been completed successfully.

Each phase typically has some associated signals that have special meanings. In the present invention, the Snoop phase and the Deferred phase or the Deferred Reply transaction are the most relevant.

The snoop signal group includes the following signals: HIT#, HITM#, DEFER#, VSBL#. These signals provide snoop result information and transaction completion information to the host bus agents in the Snoop Phase. These signals are typically generated by the chipset and are available on the host bus. The significance of these signals is explained below.

HIT# is the Cache Hit signal. It is used by the requesting agent to determine the permissible cache state of the line. The HIT# signal is also used in conjunction with the HITM# signal to indicate whether a bus agent requires a snoop phase stall.

HITM# is the Hit Modified signal. It is used to indicate whether the transaction hits a modified cache line so that the snooping agent performs an implicit write back to update the main memory. The HITM# signal is also used in conjunction with the HIT# signal to indicate whether a bus agent requires a snoop phase stall.

DEFER# is the Defer signal. It is used to indicate whether the transaction will be committed for completion immediately or if the commitment is deferred.

VSBL# is the Guaranteed Visibility signal. It may be asserted with the DEFER# signal by the deferring agent to indicate that visibility of the transaction is guaranteed for future transactions even though the transaction completion is deferred.

The visibility of the transaction indicates that the transaction is completed as seen by other transactions. An understanding of the concept of visibility is important to appreciate the role of the VSBL# signal. To illustrate this concept, an example of a system having two processors, P1 and P2, is now given. A read to address A by processor P1 is considered visible from the point of view of processor P2 if a write to A issued by P2 cannot change the value read by P1. A write to A by P1 is considered visible from the point of view of P2 if a read to A issued by P2 cannot return the value of A prior to the write by P1.

The chipset asserts VSBL# when it can guarantee the sequential order of transactions. This early indication of visibility allows the requesting agent to start future ordered transactions, since the deferred transaction is considered complete by software.

Deferred phase or Deferred Reply transaction signals are used by a deferring agent to complete a previously deferred transaction. The defer signal group includes the following signals: IDS# and ID [7:0]#. The assertion of IDS# defines the beginning of the deferred phase. The identification (ID) of the deferred transaction is contained in the ID [7:0]# signals.

In the present invention, if visibility of a deferred transaction is guaranteed, i.e., VSBL# is asserted, the Request phase for the next ordered transaction may begin. Without the visibility indication, the Request phase for the next ordered transaction may not begin until the completion of the Deferred phase or the Deferred Reply Transaction.

Snoop Results:

On observing a Request Phase for a memory access, all caching agents are required to perform an internal snoop operation and appropriately return HIT# and HITM# in the Snoop Phase. The results of combinations of HIT#, HITM#, DEFER#, and VSBL# signal encodings during a valid Snoop Phase is shown in FIG. 2.

Referring to FIG. 2, the results of the Snoop Phase include the following:

CLEAN: This result indicates that the cache line will be invalid in the caches of all other agents on the host bus. The requesting agent can store the line in any state (Modified, Exclusive, Shared, or Invalid). This result is produced by HIT# inactive, HITM# inactive, and DEFER# inactive, regardless of the state of VSBL#. The transaction is complete and the next ordered transaction is allowed.

SHARED: This result indicates that the cache line will remain valid in the cache of an agent on the host bus. The

requesting agent can store the cache line in some states (Shared or Invalid). This result is produced by HIT# active, HITM# inactive, and DEFER# inactive, regardless of the state of VSBL#. The transaction is complete and the next ordered transaction is allowed.

MODIFIED: This result indicates that the cache line is in the modified state in an agent on the host bus. This result is produced by HIT# inactive and HITM# active, regardless of the state of DEFER# and VSBL#. The transaction is complete and the next ordered transaction is allowed.

STALL: This result indicates that an agent on the host bus is not yet ready to provide a snoop result. The Snoop Phase will be stalled for some additional clocks. Any agent on the bus may use the STALL state on any transaction as a stall mechanism. This result is produced by HIT# active and HITM# active, regardless of the state of DEFER# and VSBL#. The transaction is not complete and the next ordered transaction has to wait.

DEFERRED: This result indicates that the requesting agent will not complete the transaction or make final cache state transition. This result is produced by HITM# inactive and DEFER# active, further qualified by VSBL#, regardless of the state of HIT#. The transaction is not complete. If VSBL# is inactive (while HITM# is inactive and DEFER# is active) the next ordered transaction has to wait to be issued. If VSBL# is active (while HITM# is inactive and DEFER# is active) the next ordered transaction is allowed to be issued.

The VSBL# signal, when asserted, provides an early indication that the transaction can be complete in order so that the requesting agent can start the next ordered transaction. By overlapping the preparation of the next ordered transaction and the deferred delay time, the overall throughput can be improved. If VSBL# is deasserted, the requesting agent should not issue any future order-dependent transactions until the deferred transaction is successfully completed.

Memory accesses on the host bus may be non-cacheable or cacheable. The VSBL# signal provides early indication of transaction completion in both non-cacheable and cacheable accesses.

Non-cacheable Accesses:

For non-cacheable accesses, the assertion of the VSBL# signal allows pipelining of reads and writes in the system. In systems with interconnect/ multiple paths where write transactions are typically deferred, the assertion of VSBL# guaranteeing visibility improves the overall system bandwidth significantly.

Cacheable Accesses:

The advantages of the VSBL# cacheable accesses include:

Memory access latency is reduced: Although consecutive memory accesses can be speculatively loaded into the internal cache (the L0 cache), without the VSBL# being asserted, the internal cache needs to be re-accessed. The VSBL#, when asserted, eliminates the need for re-accessing the internal RAM because the processor will proceed as normal. The use of the VSBL# signal, therefore, reduces the memory access latency.

Internal cache bandwidth is increased: The internal cache does not need to be re-accessed, resulting in higher bandwidth.

Referring to FIG. 3, a flowchart illustrating the sequence of operations in various phases is shown. For clarity and in the interest of economy, only Arbitration, Request, Error, and Snoop Phases are described.

At the start, the requesting agent first determines if it owns the bus (Step 210). If the requesting agent does not own the bus, it enters the Arbitration Phase to arbitrate for bus ownership (Step 215). In one embodiment, the arbitration occurs among four symmetric agents, and one priority agent. The symmetric agents arbitrate among themselves with a fair, round-robin scheme. The priority agent overrides any symmetric agent for request bus ownership.

The requesting agent then determines if it has obtained bus ownership (Step 220). If not, the requesting agent continues the arbitration phase. If it obtains the bus ownership, the requesting agent enters the Request Phase. In the Request Phase, the agent, now becoming the bus owner, drives the request and address information on the bus (Step 225). In one embodiment, the Request Phase is two clocks long. In the first clock, the signal ADS# (Address Strobe) is driven along with transaction address and sufficient information to begin snooping and memory access. The second clock contains the byte enable, identification (ID) information, information about the length of the transaction, and other transaction information.

The agent next enters the Error Phase. In the Error Phase, the agent reports if there are any parity errors by the request (Step 230). In one embodiment, the Error Phase occurs three clocks after the first cycle of the Request Phase. If an error is indicated, the transaction will not have any further phases. If there are no errors, the agent enters the Snoop Phase.

In the Snoop Phase, it is determined if the transaction is deferred (Step 235). If the transaction is not deferred, it is determined if stall is desired (Step 240). If stall is desired (because the agent is slow for example), control returns to step 235. If the transaction is not deferred and stall is not desired, then the transaction is complete (Step 265) and the next ordered transaction is allowed (Step 280). The agent then enters the Response Phase.

If the transaction is deferred, it is determined if sequential order is guaranteed (Step 245). If sequential order is not guaranteed, then the VSBL# signal is deasserted (Step 250), the transaction is not complete (Step 255) and the next ordered transaction is not allowed (Step 260). The agent then enters the Response Phase. If the transaction is deferred and the sequential order is guaranteed, the VSBL# signal is asserted (Step 270). The transaction is not complete (Step 275) because it is deferred but the next ordered transaction is allowed (Step 280). The agent then enters the Response Phase.

Referring to FIG. 4, a timing diagram illustrating the relationships of the signals in the Snoop and deferred phases.

At T1, the agent enters the Request phase by asserting ADS#. In one embodiment, the request information is driven on the bus for two clocks. AERR# is inactive indicating that there is no error in the Error phase. The Snoop phase begins in T5, four clocks after the assertion of ADS#. The transaction cannot be complete in order, so DEFER# is asserted at T5, HIT# and HITM# are de-asserted. The visibility of the deferred transaction is guaranteed, therefore, VSBL# is asserted at the same time with DEFER# asserted to indicate that the next ordered transaction is allowed to proceed.

At T7, the Response phase starts with the assertion of DBSY# and RS[2:0]# is driven. At T8, the IDS# is asserted